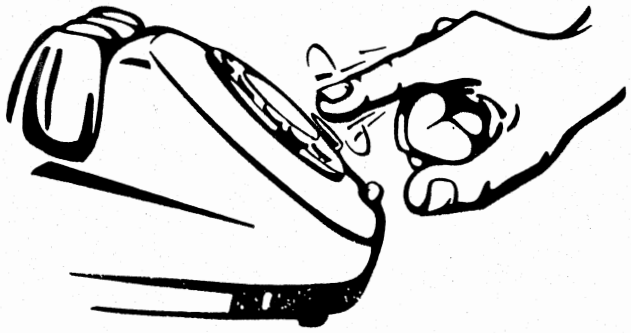
NEED HELP?



CALL US!

If you experience difficulty with this product, please call our TECH SUPPORT LINE.

(205) 881-6005

You may also contact us on INTERNET at:

support@hmi.com

or transfer files to our FTP Server on INTERNET at:

ftp.hmi.com

We also maintain a Bulletin Board which may used for file transfer purposes at:

(205) 881-7395

HUNTSVILLE MICROSYSTEMS, INC.
P.O. Box 12415
Huntsville, AL 35815

HMI APPLICATION NOTE

Getting Started With HMI Emulators and SourceGate II

Introduction

The most difficult step in using an emulator is the initial setup. The highly configurable nature of some of the newer microprocessors make this step even more challenging. The good news, however, is that once your emulator is setup to match your target configuration, you do not have to worry with it again.

This application note attempts to briefly explain some of the common stumbling points encountered during the setup of the SourceGate II debugger and during the configuration of emulator hardware, so that you can avoid them. The manual will have the details. Also, *please check the appendix of the manual and the help file* to see if specific help or guidelines are provided for the emulator that you are using. If you have any questions or problems, please contact technical support.

Configuring Software

Physical Connection

The emulator communicates to the host via serial interface or an optional Ethernet interface. The serial communications interface of the emulator supports baud rates up to 115.2 Kbaud. The rate you can actually run at will depend on the serial interface in your host. In newer PCs, serial speeds up to 115.2 Kbaud are supported, and in most UNIX workstations at least 38,400 baud is supported. The default baud rate for SourceGate II is 9600 and the default serial port is COM1 for PCs and /dev/tty00 on UNIX workstations. The faster you can operate the serial communications, the faster the emulator interface will appear (single stepping, trace display, downloading, etc.).

The emulator also supports a 25-pin parallel download port. It will connect to any Centronix interface port. You can specify a parallel download in the File-Download dialog or by using the *RDP* instead of the *RD* command on the command line.

SourceGate II under Windows

During the SourceGate II install procedure, you were prompted to select the desired COM port and baud rate. The install procedure then built a Windows icon with the necessary properties to implement your selections. If you wish to change your initial settings, you can do so by editing the Command Line field in the File-Properties dialog for the icon.

To start SourceGate II at a higher baudrate, use the "-baudrate" option: *SG2.EXE -57600*. If you change the baudrate, be sure to press the reset button on the emulator or cycle power so that the emulator will auto-detect the new baudrate. Use the "-COMn" option to specify COM2: *SG2.EXE -57600 -COM2*.

Reliable High Speed Communications under Windows

Because the standard serial communications driver supplied with Windows 3.1 does not implement hardware handshaking, serial communications beyond 9600 baud can be unreliable. To remedy this, SourceGate for Windows comes with two new serial port drivers (one for standard mode and one for enhanced mode). These drivers implement RTS/CTS handshaking to insure reliable communications up to 57600 baud under Windows 3.1. If you selected to do so, the installation procedure copied the drivers to the correct locations and modified the Windows 3.1 *SYSTEM.INI* file for you. The standard mode driver, *HMICOMM.DLL*, should be in the same directory as *SG2.EXE*. The enhanced mode driver, *HMICOMM.DRV*, should be in the *WINDOWS\SYSTEM* directory. The comm driver entry in the *WINDOWS\SYSTEM.INI* file should read comm.drv=hmicomm.drv.

Windows 95 does not generally exhibit communications problems at higher baud rates. You should be able to run up to 115.2K baud reliably when using Windows 95 in conjuction with 16550 UARTs. **NOTE: Do NOT use the HMI supplied drivers (discussed above) with Windows 95 as your system will not function properly.**

Common Communications Errors

If you receive a "The communications software has failed ..." when trying to start SourceGate or get consistent "Comport Overrun" warnings while running SourceGate, here are the most likely places to look:

a) The emulator is not connected to the COM port specified on the SourceGate II command line. As a general rule, but by no means guaranteed, COM1 is a male 9-pin connector; and COM2 (if present) can either be a 9- or 25-pin male connector. Parallel ports are almost always 25-pin female connectors.

b) Some other driver, like an old mouse driver, has been loaded on the COM port. Check *CONFIG.SYS* and *AUTOEXEC.BAT* to make sure no unwanted drivers are installed.

c) There is not enough memory is available to SourceGate. A good way to verify that this is the problem, is to start the Windows Terminal emulator (usually in the Accessories program group) and set the desired COM port for 9600 baud, 8 data bits, 1 stop bit, and no parity. Turn the emulator on or hit the emulator reset button, then hit the period key on the keyboard. If you see the emulator's initialization message followed by the "-" (dash) prompt in the terminal window, you know that the physical connection is at least operating at 9600 baud. Now try starting SourceGate at 9600 baud. If this fails, you know you probably have a PC memory (not enough) problem. Many large DOS applications, including the DOS-based SourceGate, required Expanded memory. Like most medium-to-large Windows applications, SourceGate II uses Extended memory instead of Expanded. If your have previously configured your system to use most of your memory above 640K as Expanded, you should reduce the expanded memory allocation to a minimum; this will increase available Extended Memory. Expanded memory is usually configured in your *CONFIG.SYS* file with a memory manager such as EMM386 or QEMM. Extended memory is usually enabled with HIMEM.SYS in *CONFIG.SYS*.

d) The HMI communications drivers are not properly installed. If you do not use the HMI supplied drivers (Windows 3.1), 9600 or 19200 baud is probably the maximum safe speed.

e) The serial card in your PC cannot reliably support the selected baud rate. Older PCs and PCs with less sophisticated serial communications hardware often used the older 8550 serial chip. Newer PC serial hardware uses the 16550 or 16552 chip which can buffer serial data. If you have trouble running error-free at the higher baud rates, it may be well worth the nominal cost ($30-$50) to get a new 1655x-based serial card for your PC. **Note: Windows 3.1 cannot reliably support baud rates > 57600 baud.**

SourceGate II under UNIX

The UNIX distributions for SourceGate II include shell scripts, SGSERIALA and SGETHER, that contain default baudrate and device options. If you want to use different options, you can edit the files or start SourceGate from the command line.

To start SourceGate II at a higher baudrate, use the "-baudrate" option: *SG2 -38400.* If you change the baudrate, be sure to press the reset button on the emulator or cycle power so that the emulator will auto-detect the new baudrate. Use the "-device" option to specify a different serial port or to specify the network hostname that you have defined for the emulator :

SG2 -38400 -/dev/ttyb	for serial device *ttyb*
SG2 -ihmi68360	for network emulator *hmi68360* (in /etc/hosts or yphosts)

Setting the Network Parameters on Ethernet Emulators

The first thing to do is to have your network system administrator setup/provide three things:

1) You will need to get a hostname and host IP address assigned/setup for the emulator.
2) You need to get the subnet mask for your network.
3) You need to know the IP address(es) of any routers that are local to the emulator and needed to communicate between the emulator and the workstation(s) that will run SourceGate.

To set the emulator's network address information, you will need to connect a serial line between the RS-232 port on the emulator and a dumb terminal or some kind of terminal emulator program running on your workstation or on a PC. Set the terminal options for 9600 baud, 8 data bits, 1 stop bit, and no parity. Turn the emulator on or hit the emulator reset button, then hit the period key on the keyboard. You should see the emulator's initialization message followed by the "-" (dash) prompt. At the dash prompt, type the *set_ip* command. You will be shown the Ethernet hardware address and the current IP address. You will then be prompted for several network address inputs:

- At the first prompt, you should enter the IP address of the emulator that was defined (by the network system administrator) in /etc/hosts or in yphosts. A typical IP address might look something like 192.200.4.1.

◆ The next prompt allows you to enter the subnet mask for your network. The subnet mask is used to determine if the emulator and workstation that you run SourceGate from are on the same subnet. Different subnets are usually connected by routers. Subnet masks have the same format as IP addresses; a typical subnet mask is 255.255.240.0.

◆ After entering the proper subnet mask, you will be prompted for three router IP addresses. Using the subnet mask, the emulator will identify if the it is on the same subnet as the workstation that is running SourceGate. If the emulator and workstation are not on the same subnet, the emulator will need to know the IP address of the local router that connects the subnets. The emulator can store up to three router addresses.

If you are sure that the emulator and the workstations running SourceGate will always be on the same subnet, you can just hit <return> at all 3 router address prompts, and leave them at their default value (0.0.0.0). Otherwise, enter the address or addresses of the local (local to the emulator) router(s) that connect the emulator's subnet to the subnets of any workstation that will run SourceGate.

After you enter all the network parameters and return to the dash prompt, cycle power on the emulator and allow about 15 seconds for the network communications software to boot up. You should now be able to access the emulator from the network at the new IP address.

For additional information, see HMI Application Note ACEE-A-100: *Configuring An Emulator For Ethernet Operation.*

Source-level Debug

In previous version of SourceGate, you had to explicitly enable source-level debug each time you download your absolute file (with symbols). You had to use either the *:=m* (mixed-mode) or *:=s* (source only) command to get access to your source code files for display of source code in the trace or to use the source debug window. In SourceGate II, you do not need to use these commands to use the source debug window (Source-Modules...) or to see intermixed source and assembly in the trace display.

If, however, you want to view source-only in the trace display, you need to issue the *:=s* command after loading your code and/or symbols. If you want to view intermixed source and disassembly when you issue the list (l) command, you should use the *:=m* command after loading your code and/or symbols. Either of these will cause SourceGate to align source line symbols with a source code file.

If SourceGate reports "No line number symbols found", make sure the line symbol format in the source code interface menu (Config-Source Code Interface...) matches the line number symbol that your compiler generated. If it does not, change it, reload the file, and re-issue *:=m* or *:=s*. If you still get no line symbols found, use the *:=d* command to dump a module list. If no line numbers show up for the expected modules, you may need to rebuild your absolute with the proper debug options so that line number symbols will be included in your absolute file.

Configuring Hardware

Clock Source

In the configuration window for every emulator, you are asked to select whether the processor clock will be selected from the target system or the emulator. In general, when you are plugged into your target, you should choose the target clock. There are a couple factors to consider when selecting the target clock signal:

◆ If your target processor is driven directly by an unbuffered crystal (8051, 68HC11), the capacitance of the emulator cable may prevent a proper clock signal from reaching the processor in the emulator.

- Some emulators allow you to place a matching crystal on the emulator board so that the cable loading is no longer a factor.
- If the clock frequency of the emulator clock matches that of your target clock you might just choose to run with the emulator selected as clock source.
- As a somewhat extreme measure, if the cable capacitance is loading the crystal and you cannot put a matching crystal on the emulator board, try removing the crystal capacitors from your target.

◆ Some of the higher speed emulators offer the option to buffer and clean up the target clock signal before it is driven up the clock cable. You can use the *BF* command to turn the clock buffer on. The *UB* command will unbuffer the clock; the timing skew caused by buffering might cause errors on the target hardware that is sensitive to clock-edge timing -- particularly on 68040 or 68030 targets with synchronous termination.

Memory Cycle Termination

The memory termination line provides the signal to the processor to indicate that the memory system has completed a read or write. This signal typically has a name such as READY, DTACK, DSACK, or TA. In the configuration window, most emulators ask you to specify where the memory termination signal will come from: *Emulator*, *Target*, or *According to Memory Map*. The best choice is usually *According to Memory Map*.

Control Signals

Each emulator gives you the ability to enable and disable selected target processor control signals either in the configuration menu or with individual commands in the command window.

◆ In the configuration menu you can usually specify whether the control signals coming from the target are *Disabled*, *Enabled During Emulation*, or *Always Enabled*. Typically, for running in your target, *Enabled During Emulation* is the best option. This means the emulator will only look at these signals while running target code. When the emulator is stopped, the signals will be ignored. If your target requires that these signals are always active (such as a system that accomplishes DRAM refresh with DMA), select *Always Enabled*.

◆ Examples of individual commands are the *EH* and *NH* (Enable Halt and No halt) commands for the 68XXX series of emulators.

◆ **Target System Reset**. If your target system has active circuitry that pulls the Reset line down strongly, you may need to leave the Target System Reset selection as "Disabled". You will know if the Reset "Enabled" selection is a problem if you immediately get an "Emulator is Running" message. If this is the case, you will need to use the CONFIG command to regain control of the emulator.

If you do need to operate with target reset disabled, this just means that the processor will not respond to a target board reset signal. You can use the RS or RP command to reset the processor instead.

Shadowed Target ROM

Many systems, especially 68XXX targets, relocate a block of ROM memory from address 0 to some other address immediately after a target reset. This lets the processor fetch its initial reset vector and stack pointer from low memory, then frees the low address space for a block of RAM that was shadowed by the ROM. Depending on how the shadow mechanism works, the emulator may not see the shadowed block. After a reset, check to see that your initial PC is what you expect. If it is not, you can manually set the PC or build a small command file to do so:

```
rs
xpc
010000
xa7
0a00000
```

68020, 68030, 68040, and 68060 Disassembly

The staged pipeline architecture of the instruction prefetch unit within the 68020/030/040/060 makes disassembly of bus activity very difficult. In order to provide more accurate disassembly for these processors, you need to execute the *OP* instruction after you have loaded your code with the *RD* command. The *OP* instruction will set a bit in a trace tag RAM at each address where an instruction starts. When the trace buffers are captured, this extra bit is used to unscramble the memory activity and display meaningful trace results. The syntax for the OP instruction is *OP beg_addr, end_addr*. You can tag up to 4 Mbytes of code in 4 blocks and the blocks do not have to be in a contiguous space.

Target Address Latching

Some targets provide advanced protection circuitry that detects data parity or memory existence at all accessed addresses. This may cause problems if the emulator's background monitor is generating address activity that falls outside valid limits for the target hardware. The monitor program is the processor code that is running when the emulator is stopped or idle. To make sure that the target always sees address activity that is valid, the 68XXX series of emulators provide the *TA* and *EL* commands. The *TA (Target Address)* command lets you specify a valid target address that the target hardware will see if the emulator is running in the monitor. Assuming that you had valid memory at 0a0000, you could specify *TA 0a0000* on the command line. The *EL (Enable Latching)* command tells the emulator to start using the latch address that you specified with the *TA* command.

6833x, 6834x, 68HC16, 68306/307, and 68360 Configuration

By far, some of the most configurable processors are the Motorola CPU16 and CPU32 integrated communications chips. Many of the target hardware specifics get programmed into the configuration registers within the chip. Often, problems that seem to be associated with using the emulator are really chip configuration problems. The 68306/307 family of devices also have potential configuration pitfalls. Here is an overview of some of the things to watch out for:

- **Emulation Memory**. There are no memory restrictions for running the emulator out of target memory; all modes are supported -- synchronous, asynchronous, 16-bit, 8-bit, with 0-n wait states. There are, however, two restrictions when using the emulation memory system as a substitute for target memory:

  - Only asynchronous accesses are supported
  - High byte and low byte access for a single 16-bit memory block must occur on a single chip select. This implies that bits 13 and 14 of CSORx must both be set to 1.

This means that you may have to alter the values in your CSOR registers if you want to use emulation memory instead of target memory during debug. From a code execution standpoint, this should have no effect.

- **Pre-Programming Internal Registers.** One of the problems with debugging with one of these chips is the fact that the internal memory configuration (chip selects, base addresses, wait states, etc.) of the chip gets destroyed each time that the chip is reset. HMI emulators have improved this situation by allowing you to specify reset values for a number of the internal configuration registers. The exact registers that you can pre-define depends on the processor; please see your manual for details. Immediately after you issue an RS command, the emulator will program these registers with the configuration values.

Once you have programmed these selections to your desired value, they will be saved in nonvolatile memory within the emulator. You can also save/recall the configuration to/from a host file. On the command line use the following:

To save: w[%] 0 *myname.cfg*

To recall: rd[%] *myname.cfg*

The registers that you can enter values for on the 6833x, 68340, and 68HC16 are the MCR, CSPARx, PxPAR, CSBARx, and CSORx.

◆ **MCR.** You may want to alter your programming of the MCR to take advantage of 2 advanced emulator features.

If you set FRZSW (bit 14) and FRZPT (bit 13) to a one, the emulator can freeze the watchdog and periodic interrupt timers when you stop or hit a break point.

If you set SHEN1 (bit9) and SHEN0 (bit 8) the emulator can display internal accesses in the trace display. These accesses will be marked with a asterisk (*).

◆ **CSPARx.** The setup of CSPAR0 is especially important if CSBOOT interfaces to 8-bit wide memory. In order for the emulator to correctly read boot memory on an RS (reset) command, the lowest 2 bits of CSPAR0 must be set to 10b.

If your target memory (especially BOOT ROM) does not use the internal chip select signals, you need to set the necessary bits in CSPAR0 and CSPAR1 to enable (01b) the address control logic that your target uses to decode memory addresses. These signals typically include the upper address lines (A19-A23) and the function codes (FC0-FC2).

◆ **PEPAR and PFPAR.** The HMI emulator does not impose any restrictions on your use of PortE pins. They can be configured as either I/O or control pins. The emulator, however, needs to know how your target uses those pins so that it can synthesize any "missing" control signals it needs for internal operations such as trace clocking. You should at least program PEPAR to the expected value.

◆ **CSBARx and CSORx.** The chip select lines are used to synthesize a DSACK equivalent if the DSACK lines are used as I/O lines. Therefore, it is important to program the CSBARx and CSORx register entries.

The values that you program into the CSBARx and CSORx registers are also important for correct address disassembly in the trace buffers. Since the chip select lines are used for upper address line decoding, the values that appear on the external processor address lines do not always reflect the true CPU memory address. If you have programmed the proper SIM value in the configuration menu, the trace disassembler can properly display the full address in the trace.

** Remember, just as with the chip itself, if you reset the emulator, the SIM config registers will be reset to their default values (or to the values you specify in the config menu). So, you may not be able to display any or all of your memory if the chip selects and option registers are not programmed correctly.

◆ **R/W Bits**. The option registers allow you to specify R/W protection on each block of memory. Typically, the CSORBT (CSBOOT option register) will denote read only protection for the boot ROM block of memory. Usually, emulation RAM is used to overlay this block of ROM so that you can modify, download, and debug code without having to burn new ROMs each time you make a change.

** FOR DOWNLOADING PURPOSES, BE SURE to change the CSOR for any ROM block to reflect full R/W permission. Otherwise, a file download will not be able to overwrite your old code. When you are ready to burn PROMs, you should change your code back to specify read only protection.

◆ **Watchdog Timer/SYPCR**. The processor's watch dog timer (WDT) can be a nuisance while trying to debug code. Specifically, the WDT can cause unexpected results while trying to single step or when you try to run again after hitting a breakpoint. The reset value for the system protection control register (SYPCR) on some chips (6833x and 68HC16) is WDT enabled. There are a couple ways to prevent WDT interference while debugging your system:

- If you do not need the WDT while debugging, write a 0x00 to the SYPCR immediately after a target reset using the EXRSYPCR command or a fill command: F(sd)7ffa20,7ffa20,00. You can also alter your initialization code to write a 0x00 to SYPCR. Remember that the SYPCR can only be written **once** after a reset.

- If you want to run with the WDT enabled, but need to set breakpoints, be sure that bits 13 and 14 of the MCR are programmed to high state. This lets the emulator freeze the WDT and PT (periodic timer) when a breakpoint is encountered.